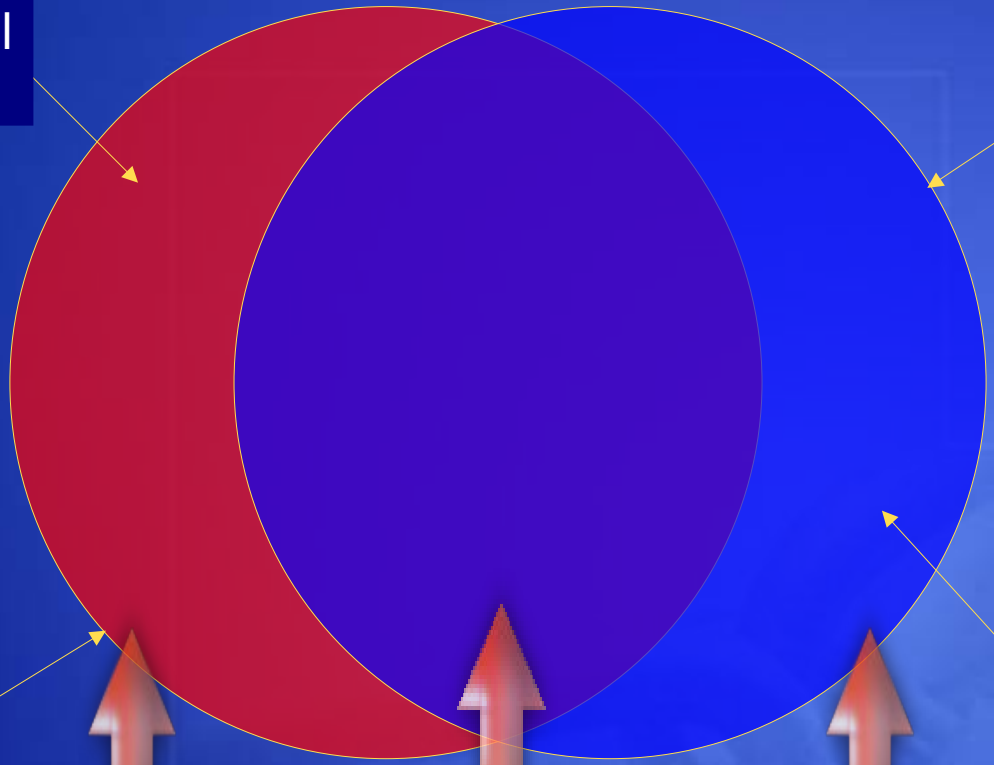
# Introduction to Security Testing

Shawn Hernan
Security Program Manager
Security Engineering and Communication

# Security Testing



Traditional faults

Actual software functionality

Intended functionality

Unintended, undocumented or unknown functionality

**Missing Defenses**

No authn

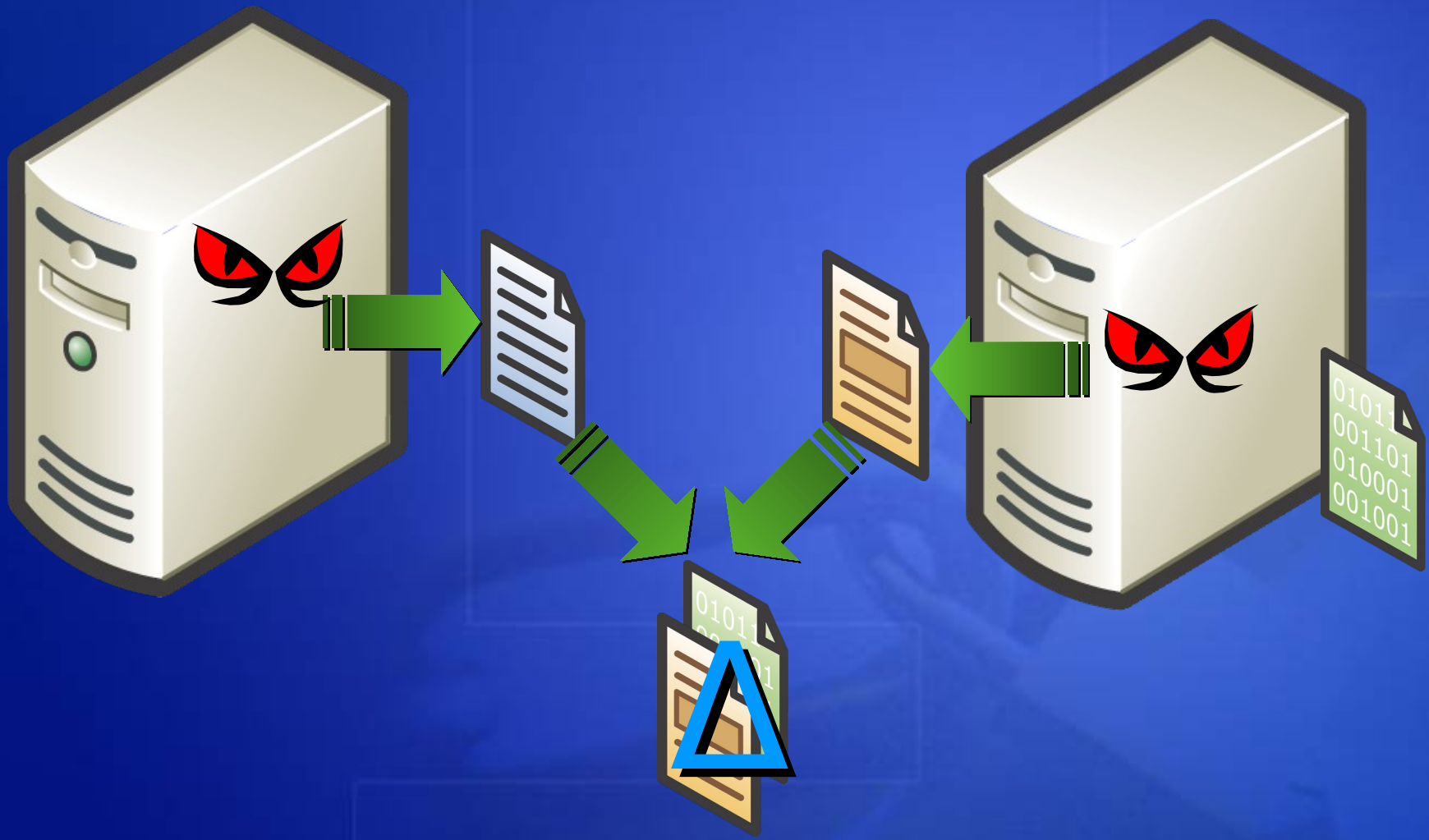
**Poor Defenses**

Weak authn

**Extra 'functionality'**

BO in authn

# Testing Like an Attacker: ‘Footprint’ the Application



# Fuzz Testing

- Fuzz Testing is the methodical application of malformed data in a search for vulnerabilities
- Find security & reliability issues efficiently

# How to Fuzz (1 of 4)

- Determine all the entry points to your code
  - Network ports and protocols
  - Files and file types
- Rank them by privilege level and accessibility
  - Anonymous, user, admin
  - Remote, local
- Run your app under Application Verifier

# How to Fuzz (2 of 4)

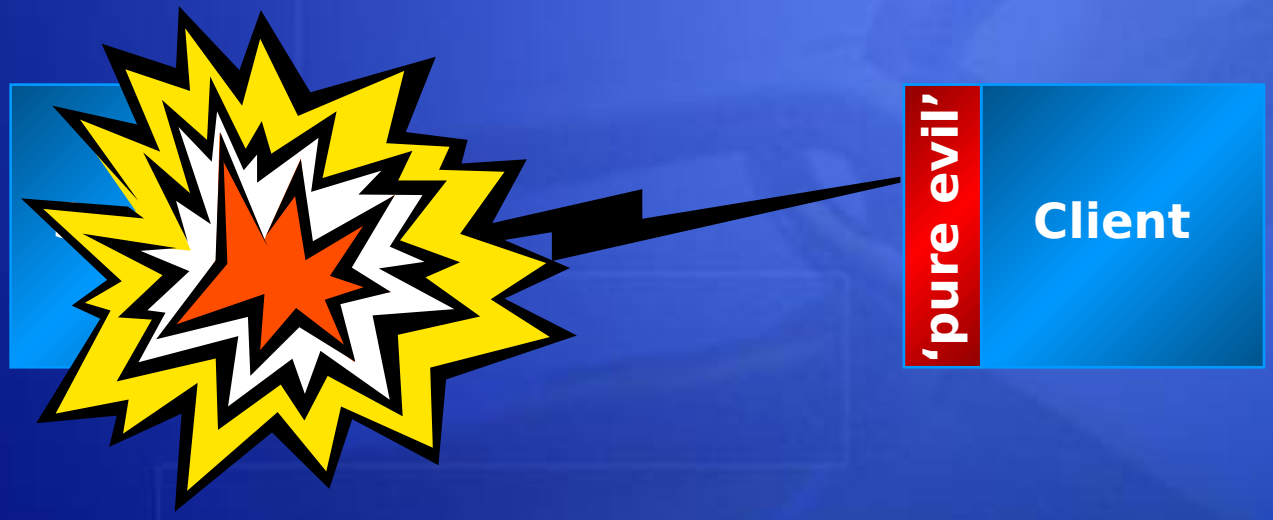
- For ALL file formats you consume
  - Build a collection of valid files
  - Tweak a file at random using a tool
  - Load the file into your application
  - Observe!
    - Crash? Memory spike?
- For all network ports
  - Use a rogue client/server

# How to Fuzz (3 of 4)

- Examples of ‘tweaking’ a file
  - Write a random series of bytes
  - Flip two adjacent bytes
  - Look for ASCII/Unicode text and then set the trailing NULL to non-NULL
  - Set size values to random numbers
  - Set integer to negative number
  - Etc…

# How to Fuzz (4 of 4)

- Network fuzzing
  - Build a rogue front-end to your app (client and server)
    - Tweak bits at random



# Attack Ideas

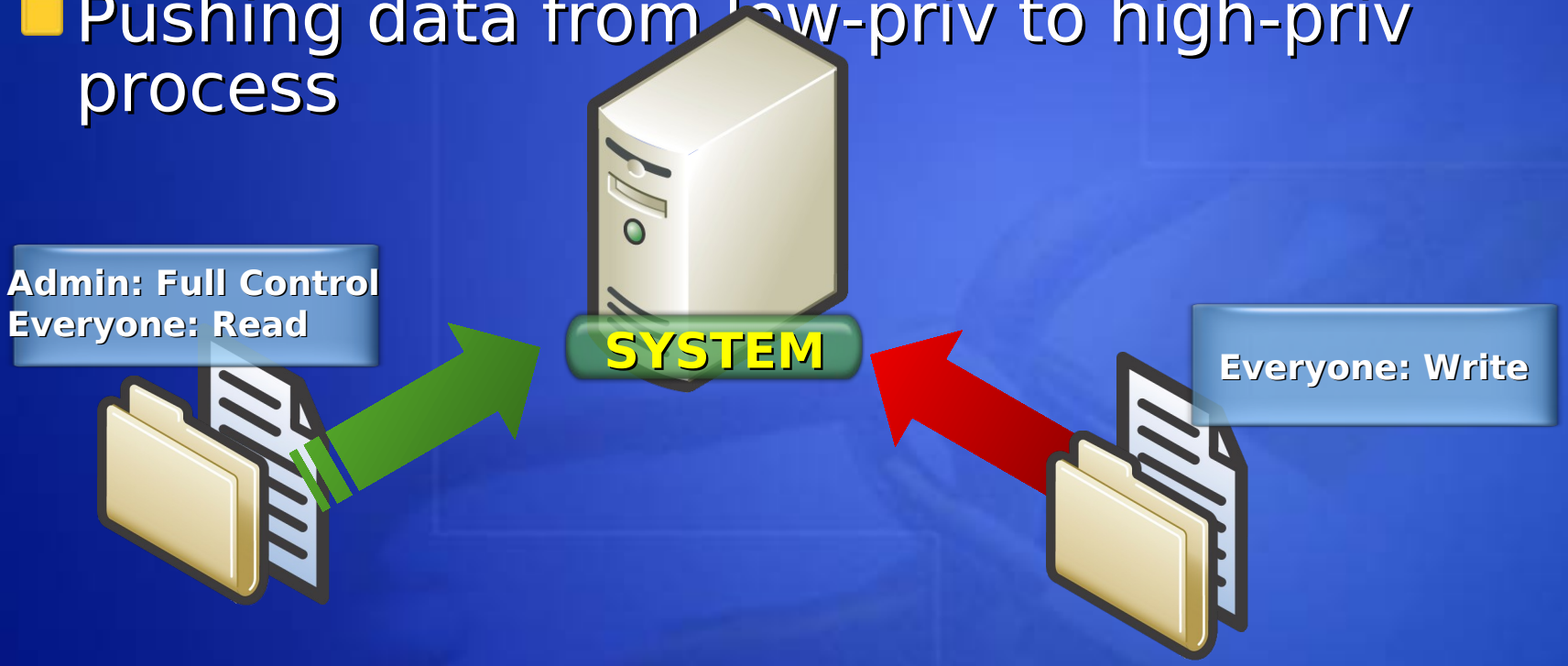
- Rule #1 – There are no rules
- If you provide a client to access the server, don’t use it!
  - Mimic the client in code
- If you rely on a specific service, build a bogus one

# “Bang for the Buck” Attack Ideas

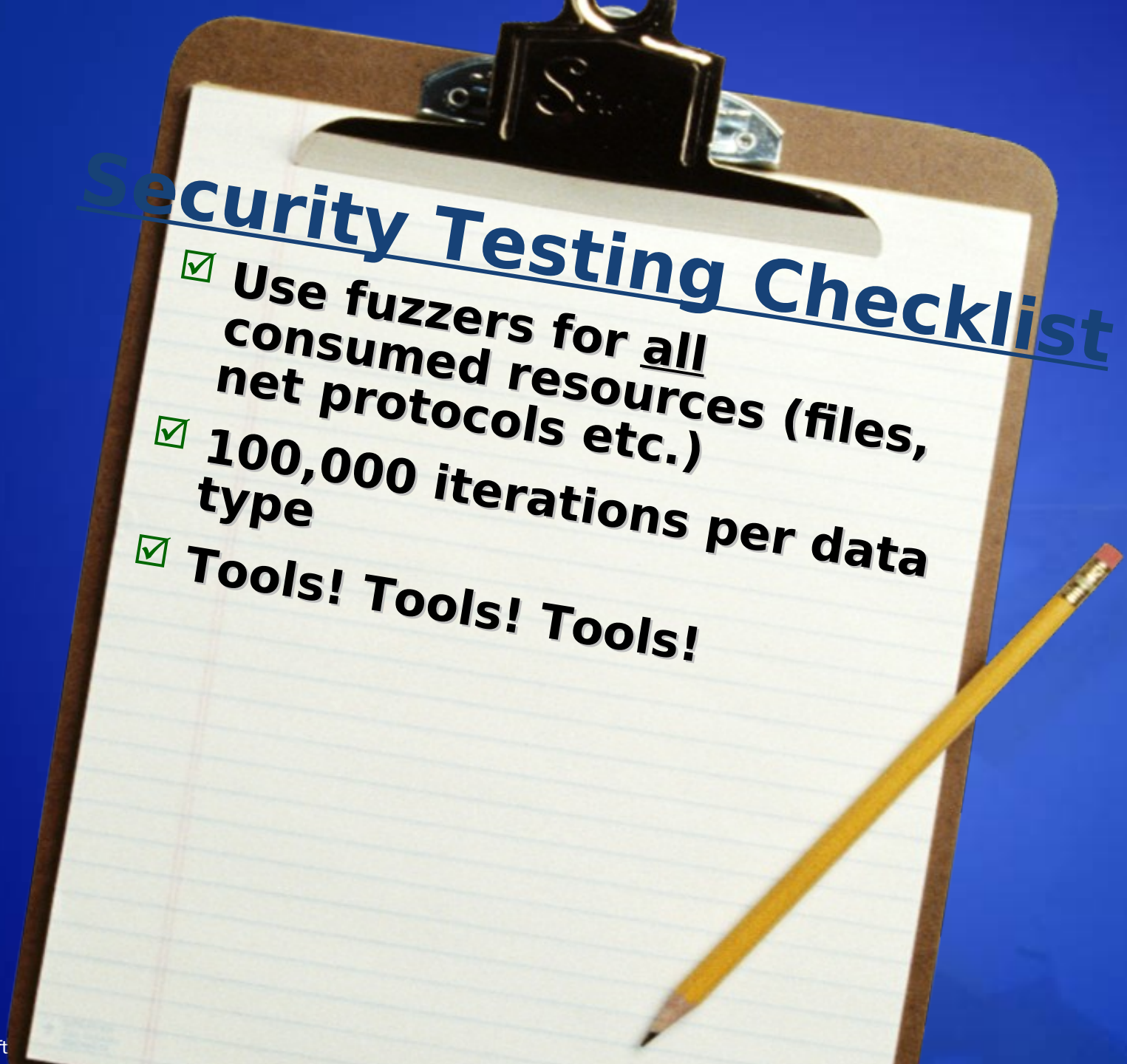
- Consume files?
  - Try device names and ‘..’
  - Look for: hangs, access to other files
  - Fuzz data structures
  - Look for: AVs or memory leaks (appverifier)
- Look for PII data in information disclosure threats
- grep for ‘should’ and ‘assume’ in the code :)
- ActiveX (especially Safe For Scripting)
  - Look at each method/property and ask,

# “Bang for the Buck” Attack Ideas

- Look for privilege-elevation boundaries
  - Pushing data from low-priv to high-priv process



# Security Testing Checklist



- ☑ Use fuzzers for all consumed resources (files, net protocols etc.)
- ☑ 100,000 iterations per data type
- ☑ Tools! Tools! Tools!